analysis

December 29, 2022

1 Suicide Data Analysis

```
[80]: # Load necessary libraries
      import numpy as np
      import pandas as pd
      from sklearn.ensemble import RandomForestClassifier, ExtraTreesRegressor
      from sklearn.inspection import permutation_importance
      from sklearn.model_selection import train_test_split
      import matplotlib.pyplot as plt
```

1.1 Feature Importance

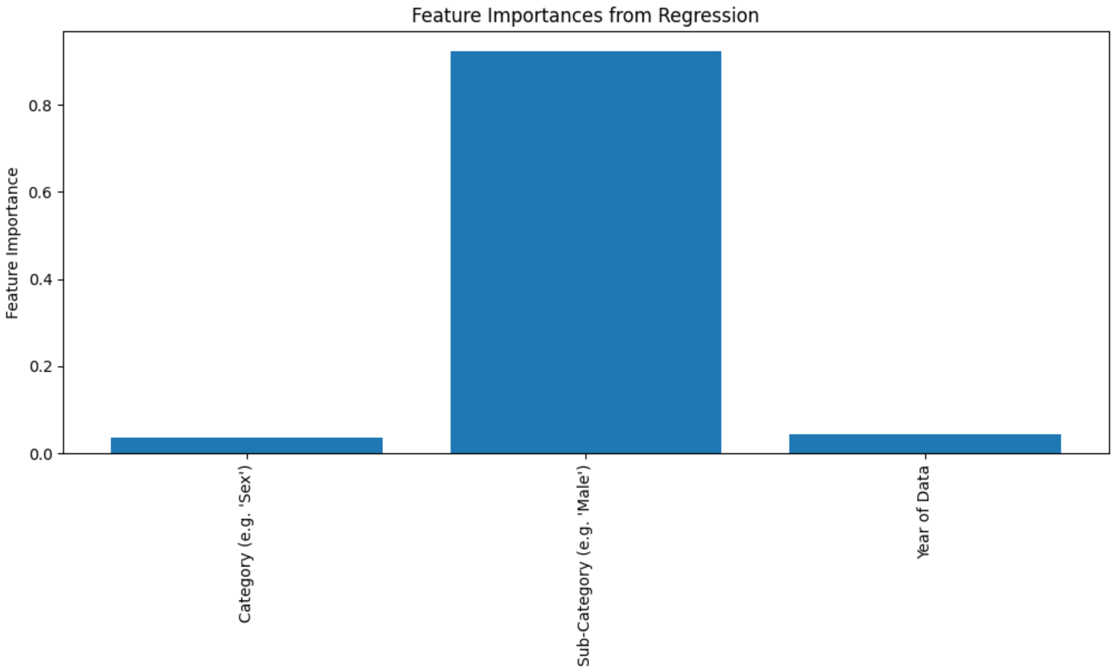
First we look at the feature importance using a regression and a random forest.

```
[81]: # Load the dataset and prepare it for modeling
      df = pd.read_csv("suicide_data.csv")
      df = df.drop(columns=['INDICATOR', 'UNIT', 'UNIT_NUM', 'STUB_NAME',␣
       ↪'STUB_LABEL', "FLAG", "AGE", "AGE_NUM"])
      numeric_cols = ['STUB_NAME_NUM', 'STUB_LABEL_NUM', "YEAR_NUM", "ESTIMATE"]
      df[numeric_cols] = df[numeric_cols].apply(pd.to_numeric)
      df = df.dropna()
      feature_names = ["Category (e.g. 'Sex')", "Sub-Category (e.g. 'Male')", "Year␣
       ↪of Data"]
```

```
[82]: # Split the data into a training set and a test set
      X = df.drop(["YEAR", "ESTIMATE"], axis=1)
      y = df["ESTIMATE"].astype(int)
      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

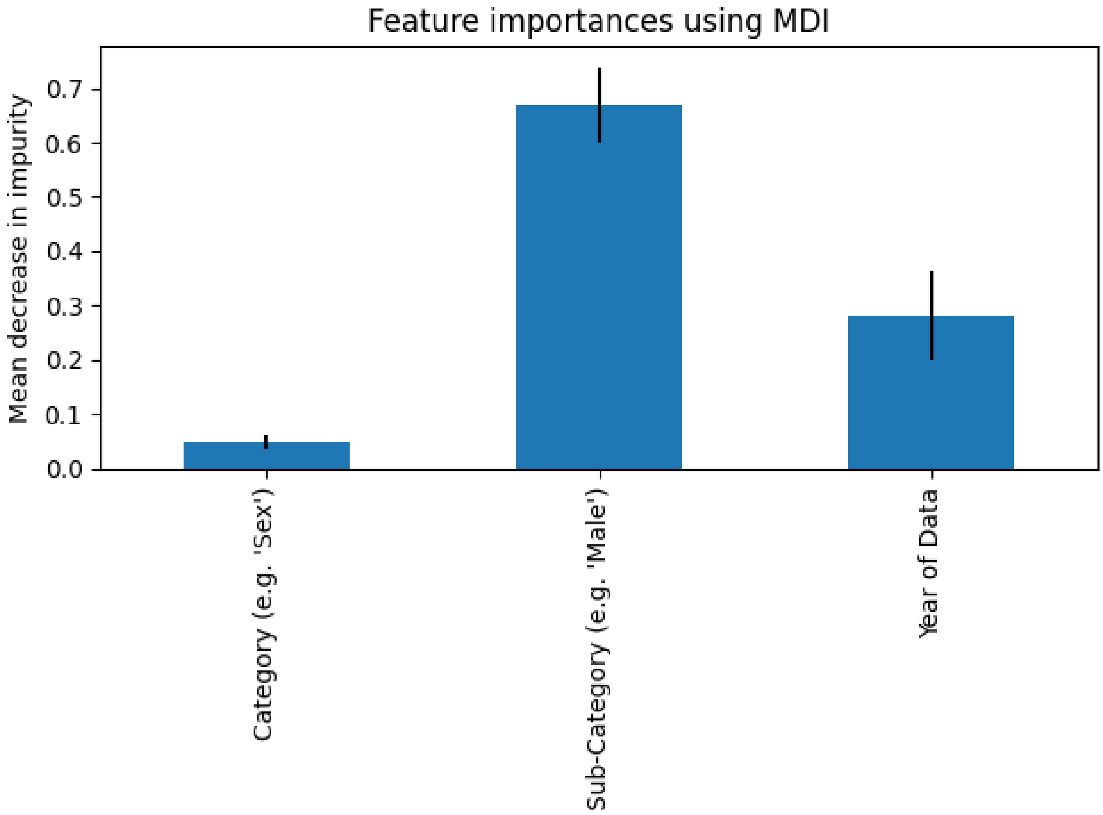
```
[83]: # regression feature analysis
      regressor = ExtraTreesRegressor()
      regressor.fit(X_train, y_train)
      importances = regressor.feature_importances_
```

```
[84]: # figure 0 (regression feature importance)
      plt.figure(figsize=(10, 6))
      plt.bar(range(len(importances)), importances)
      plt.xticks(range(len(importances)), feature_names, rotation=90)
      plt.tight_layout()
      plt.title("Feature Importances from Regression")
      plt.ylabel("Feature Importance")
      plt.savefig('feature_importances.png')
      plt.show()
```



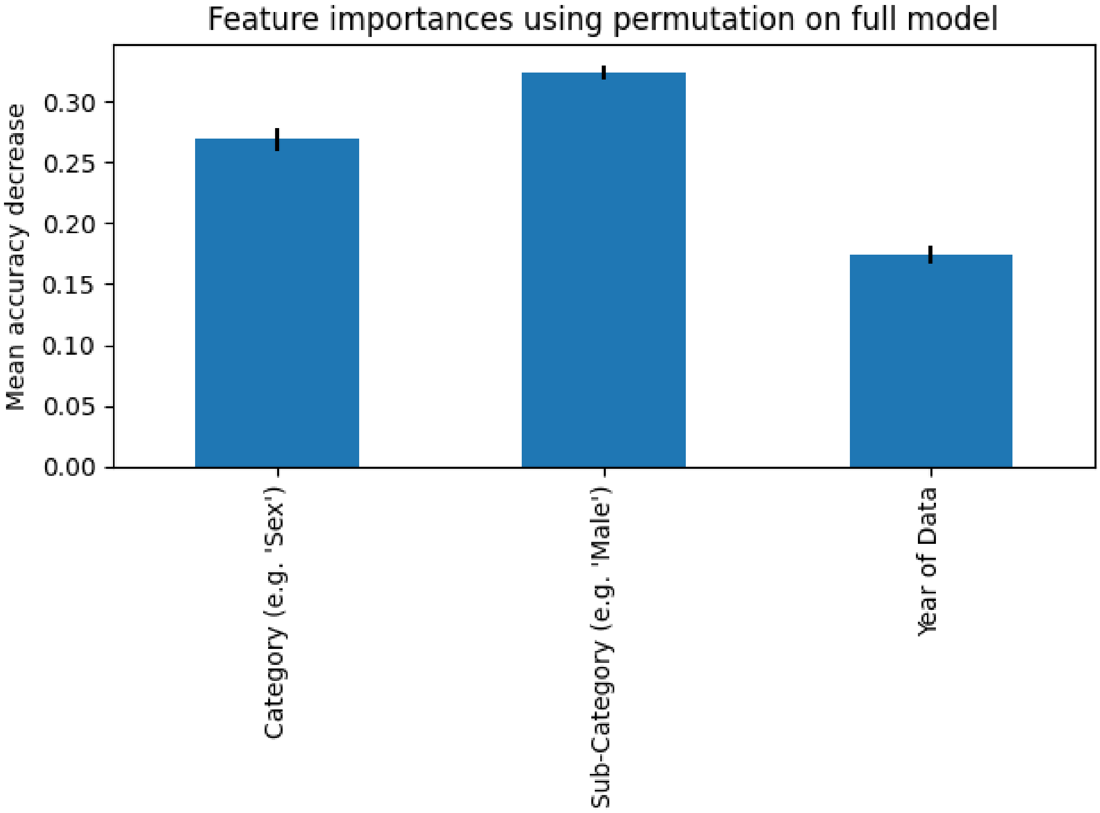
[85]:
```
# random forest feature analysis
clf = RandomForestClassifier()
clf.fit(X_train, y_train)
importances = clf.feature_importances_
std = np.std([tree.feature_importances_ for tree in clf.estimators_], axis=0)
forest_importances = pd.Series(importances, index=feature_names)
```

[86]:
```
# figure 1 (random forest feature importance)
fig, ax = plt.subplots()
forest_importances.plot.bar(yerr=std, ax=ax)
ax.set_title("Feature importances using MDI")
ax.set_ylabel("Mean decrease in impurity")
fig.tight_layout()
plt.savefig('feature_importances_1.png')
plt.show()
```



```
[87]: # Calculate permutation importance
      result = permutation_importance(
          clf, X_test, y_test, n_repeats=10, random_state=42, n_jobs=2
      )
      forest_importances = pd.Series(result.importances_mean, index=feature_names)
```

```
[88]: fig, ax = plt.subplots()
      forest_importances.plot.bar(yerr=result.importances_std, ax=ax)
      ax.set_title("Feature importances using permutation on full model")
      ax.set_ylabel("Mean accuracy decrease")
      fig.tight_layout()
      plt.savefig('feature_importances_2.png')
      plt.show()
```



1.2 Analysis and Visualization

Given that there seem to be some clearly important features, lets analyze them.

[89]:

```
# load data for graphing
df = pd.read_csv("suicide_data.csv")
df_sex = df.loc[df["STUB_LABEL"].str.contains("Male|Female"), :]
df_sex.loc[:, "Sex"] = df_sex["STUB_LABEL"].apply(lambda x: x.split(":")[0])
# Group the data by sex and find the average rate for each group
df_grouped = df_sex.groupby("Sex").mean()
df_grouped = df_grouped.reset_index()
df_sex.loc[:, "Attributes"] = df_sex["STUB_LABEL"].apply(lambda x: x.split(":
  ↪")[1] if ":" in x else "")
df_sex_mean = df_sex.groupby(["Sex", "Attributes"]).mean(numeric_only=True)
df_sex_mean = df_sex_mean.reset_index()
df_sex_pivot = df_sex_mean.pivot(index="Sex", columns="Attributes",␣
  ↪values="ESTIMATE")
```

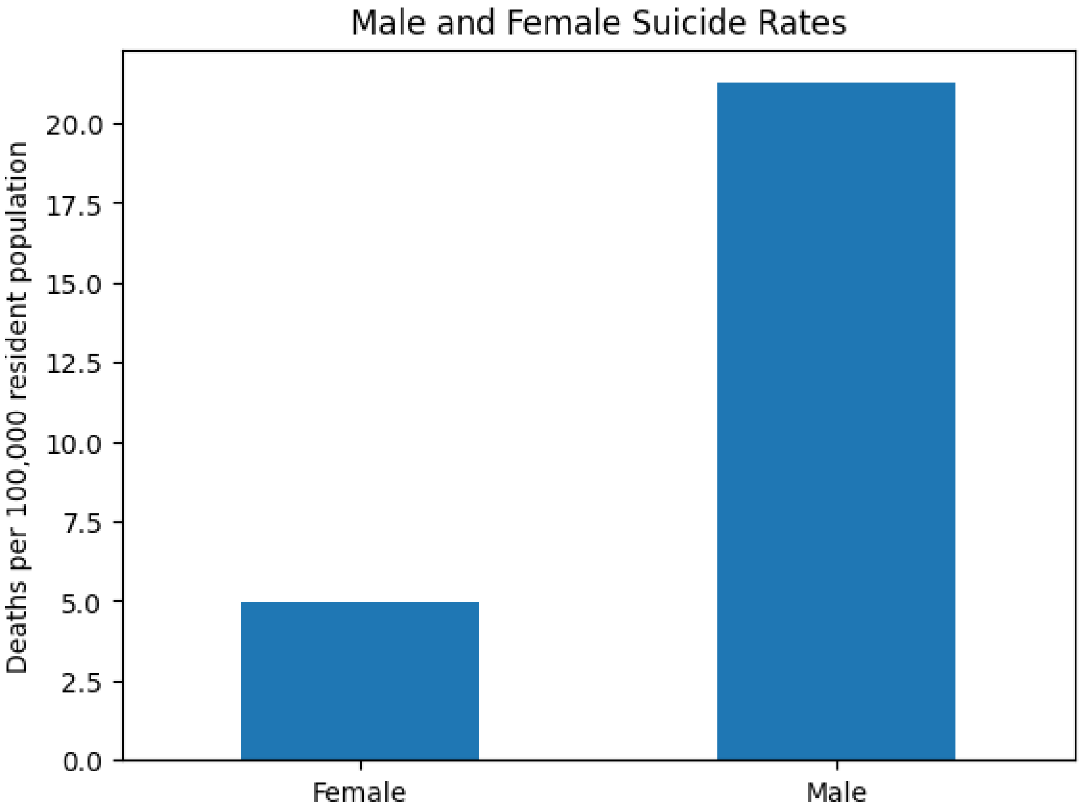
```
C:\Users\edwin\AppData\Local\Temp\ipykernel_29088\3236775883.py:4:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_sex.loc[:, "Sex"] = df_sex["STUB_LABEL"].apply(lambda x: x.split(":")[0])
C:\Users\edwin\AppData\Local\Temp\ipykernel_29088\3236775883.py:6:
FutureWarning: The default value of numeric_only in DataFrameGroupBy.mean is
deprecated. In a future version, numeric_only will default to False. Either
specify numeric_only or select only columns which should be valid for the
function.
  df_grouped = df_sex.groupby("Sex").mean()
C:\Users\edwin\AppData\Local\Temp\ipykernel_29088\3236775883.py:8:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_sex.loc[:, "Attributes"] = df_sex["STUB_LABEL"].apply(lambda x:
x.split(":")[1] if ":" in x else "")
```

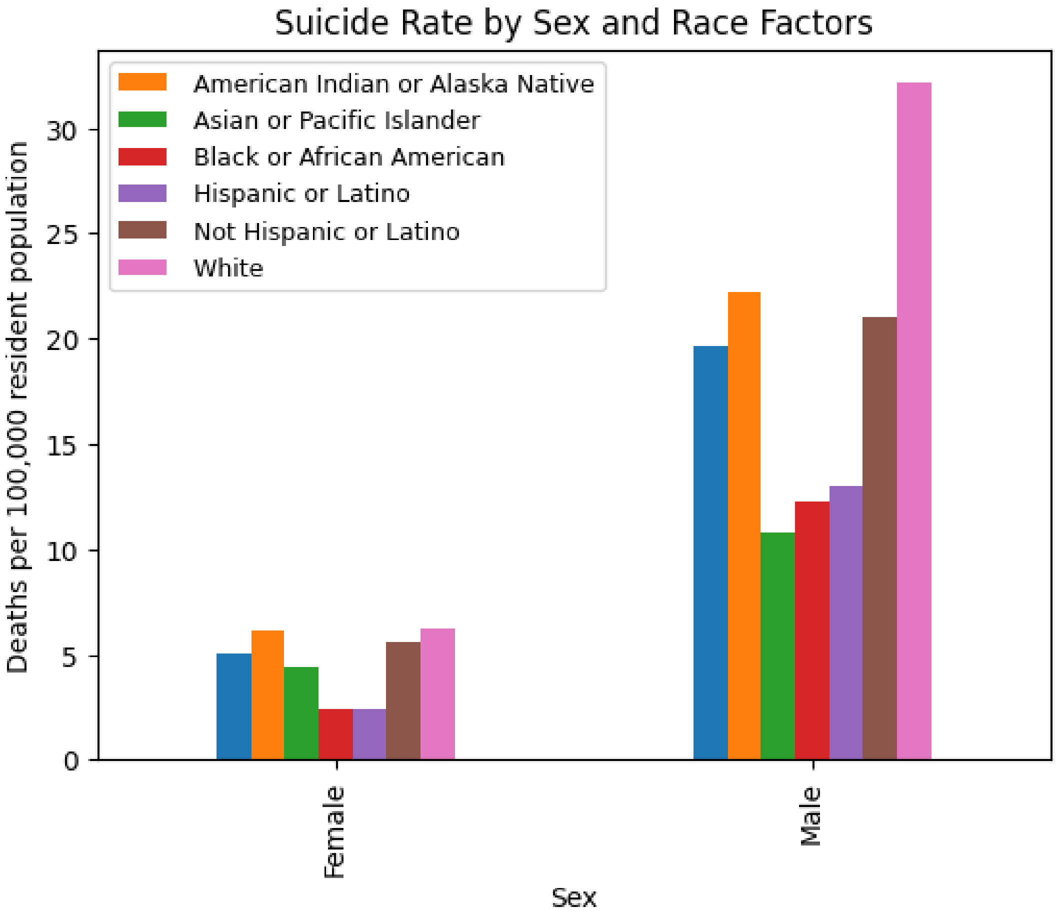
[90]:

```python
# male and female plot
df_plot = df_grouped[['ESTIMATE']]
ax = df_plot.plot.bar(rot=0)
plt.xticks(range(len(df_plot)), ['Female', 'Male'])
ax.legend_.remove()
plt.title("Male and Female Suicide Rates")
plt.ylabel("Deaths per 100,000 resident population")
plt.savefig("suicide_rates_by_sex.png")
plt.show()
```

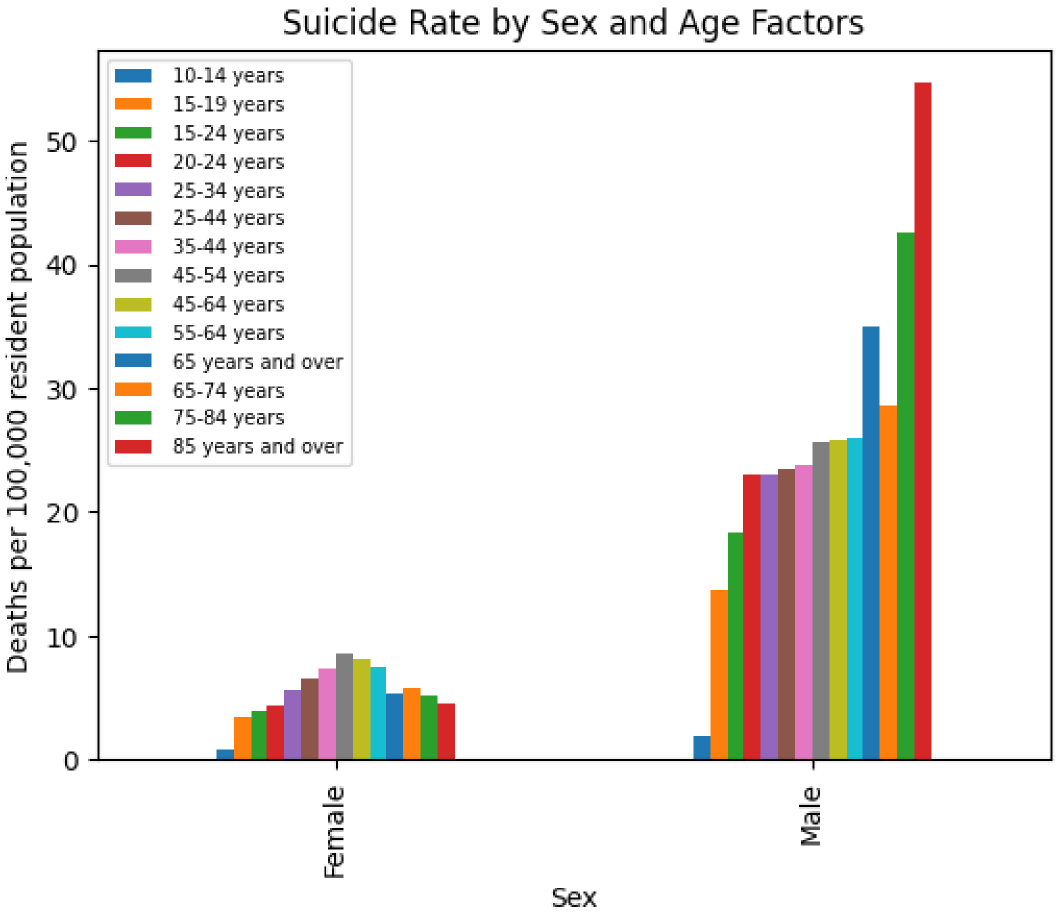


[91]:

```
# sex and race graph
race_factors = df_sex_mean[~df_sex_mean["Attributes"].str.
 ↪contains("\d+")]["Attributes"].unique()
df_race_pivot = df_sex_mean[df_sex_mean["Attributes"].isin(race_factors)].
 ↪pivot(index="Sex", columns="Attributes", values="ESTIMATE")
df_race_pivot.plot.bar()
plt.title("Suicide Rate by Sex and Race Factors")
plt.ylabel("Deaths per 100,000 resident population")
plt.legend(fontsize=9)
plt.savefig("sex_race_factors.png")
plt.show()
```



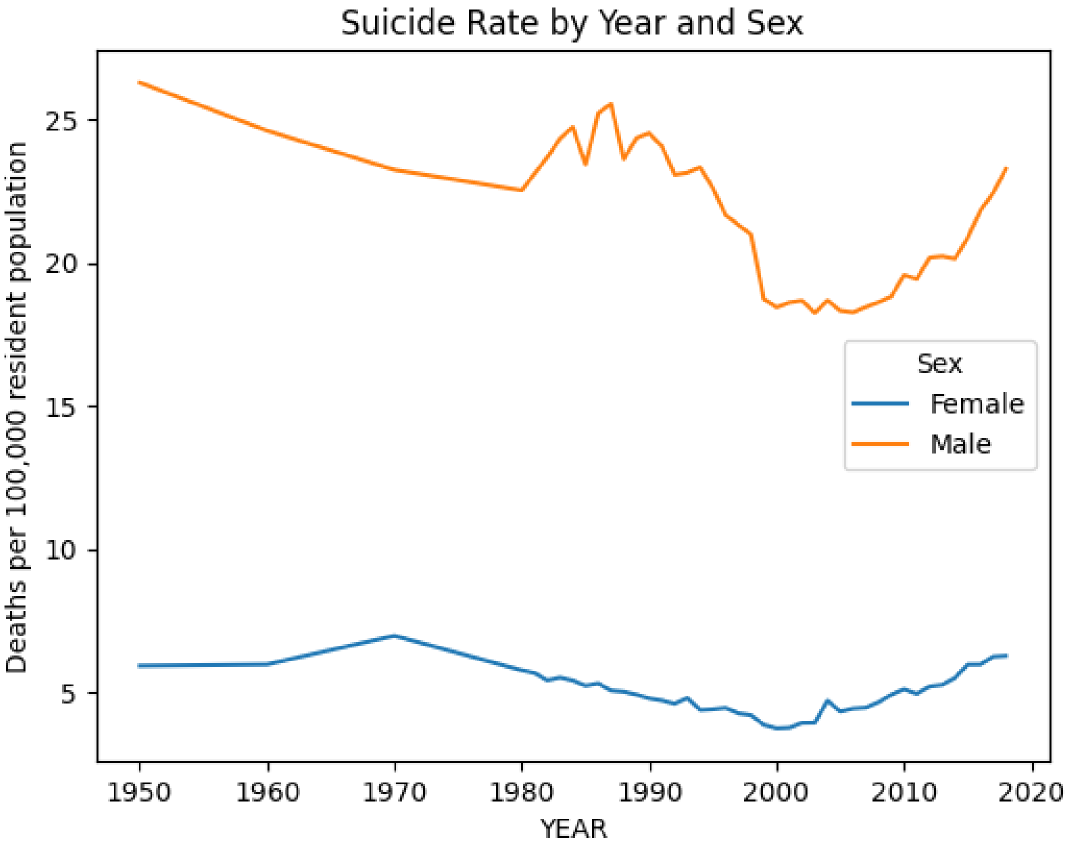
```
[92]: # sex and age graph
age_factors = df_sex_mean[df_sex_mean["Attributes"].str.
 ↪contains("\d+")]["Attributes"].unique()
df_age_pivot = df_sex_mean[df_sex_mean["Attributes"].isin(age_factors)].
 ↪pivot(index="Sex", columns="Attributes", values="ESTIMATE")
df_age_pivot.plot.bar()
plt.title("Suicide Rate by Sex and Age Factors")
plt.ylabel("Deaths per 100,000 resident population")
plt.legend(fontsize=7)
plt.savefig("sex_age_factors.png")
plt.show()
```



[93]:

```
# sex and year graph
df_sex_mean = df_sex.groupby(["YEAR", "Sex"]).mean()
df_sex_mean = df_sex_mean.reset_index()
df_sex_pivot = df_sex_mean.pivot(index="YEAR", columns="Sex", values="ESTIMATE")
df_sex_pivot.plot()
plt.title("Suicide Rate by Year and Sex")
plt.ylabel("Deaths per 100,000 resident population")
plt.savefig("year_sex.png")
plt.show()
```

```
C:\Users\edwin\AppData\Local\Temp\ipykernel_29088\4051993214.py:2:
FutureWarning: The default value of numeric_only in DataFrameGroupBy.mean is
deprecated. In a future version, numeric_only will default to False. Either
specify numeric_only or select only columns which should be valid for the
function.
  df_sex_mean = df_sex.groupby(["YEAR", "Sex"]).mean()
```


Suicide Rate by Year and Sex
Deaths per 100,000 resident population
25
20
15
10
5
1950
1960
1970
1980
1990
2000
2010
2020
YEAR
Sex
Female
Male